Reconstructing Curves from Points and Tangents

L. Greengard and C. Stucchio

March 10, 2009

Abstract

Reconstructing a finite set of curves from an unordered set of sample points is a well studied topic. There has been less effort that considers how much better the reconstruction can be if *tangential* information is given as well. We show that if curves are separated from each other by a distance δ, then the sampling rate need only be $O(\sqrt{\delta})$ for error-free reconstruction. For the case of point data alone, $O(\delta)$ sampling is required.

1 Introduction

In this paper, we consider the problem of reconstructing a C^1 *figure* – that is, a family of curves $\{\gamma_i(t)\}_{0...M-1}$ from a finite set of data. More precisely, we assume we are given an unorganized set of points $\{\vec{p}_i\}_{i=0...N-1}$, as well as *unit* tangents to the points $\{\vec{m}_i\}_{i=0...N-1}$. Note that the tangents have no particular orientation; making the change $\vec{m}_i \to -\vec{m}_i$ destroys no information.

Definition 1.1 *A polygonalization of a figure $\{\gamma_i(t)\}_{0...M-1}$ is a planar graph $\Gamma = (V, E)$ with the property that each vertex $p \in V$ is a point on some $\gamma_i(t)$, and each edge connects points which are adjacent samples of some curve γ_i.*

Our goal here is to construct an algorithm which reconstructs the polygonalization of a figure from the data defined above. An example of a polygonalization is given in Figure 1.

The topic of reconstructing figures solely from point data $\{\vec{p}_i\}_{i=0...N-1}$ has been the subject of considerable attention [3, 4, 9, 14, 5, 10, 11]. This is actually a more difficult problem, and only weaker results are possible. The main difficulty is the following; if the distance between two separate curves γ_i and γ_j is smaller than the sample spacing, then it is difficult to determine which points are associated to which curve. Thus, sample spacing must be $O(\delta)$, with δ the distance between different curves.

Tangential information makes this task easier; in essence, if two points are nearby (say $\vec{p}_1$ and $\vec{p}_2$), but $\pm\vec{m}_1$ does not point (roughly) in the direction $\vec{p}_2 - \vec{p}_1$, then $\vec{p}_1$ and $\vec{p}_2$ should not be connected. This fact allows us to reduce the sample spacing to $O(\delta^{1/2})$, rather than $O(\delta)$. This is to be expected by analogy

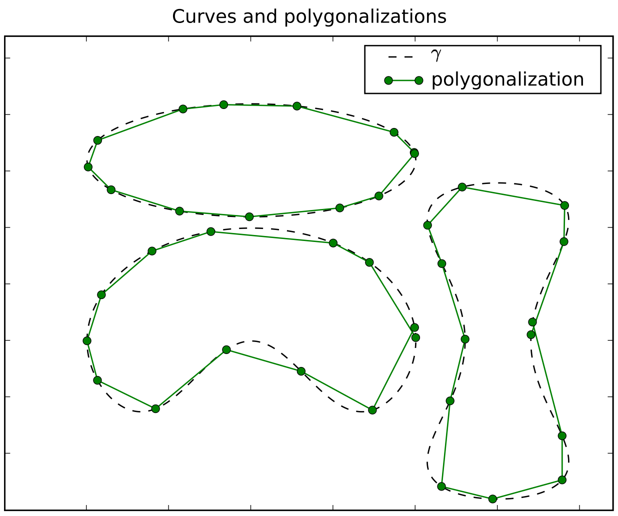

Figure 1: A figure and it's polygonalization, c.f. Definition 1.1.

to interpolation; knowledge of a function and its derivatives yields quadratic accuracy.

We should mention at this point related work on *Surfels* (short for *Surface Elements*). A surfel is a point, together with information characterizing the tangent plane to a surface at that point (and perhaps other information such as texture). They have become somewhat popular in computer graphics recently, mainly for rendering objects characterized by point clouds [1, 2, 6, 15, 17, 18].

In this work, we present an algorithm which allows us to reconstruct a curve from $\{\vec{p}_i, \vec{m}_i\}_{i=0 \ldots N-1}$. We make two assumptions, under which the algorithm is provably correct.

Assumption 1 *We assume each curve $\gamma_i(t) = (x_i(t), y_i(t))$ has bounded curvature:*

$$\forall i = 0 \ldots M-1, \; \frac{|x_i'(t) y_i''(t) - y_i'(t) x_i''(t)|}{(x_i'(t)^2 + y_i'(t)^2)^{3/2}} \leq \kappa_m \tag{1.1}$$

This assumption is necessary to prevent the curves from oscillating too much between samples.

Assumption 2 *We assume the curves γ_i and γ_j are uniformly separated from each other, i.e.:*

$$\inf_{t,t'} |\gamma_i(t) - \gamma_j(t')| \geq \delta \text{ for } i \neq j \tag{1.2a}$$

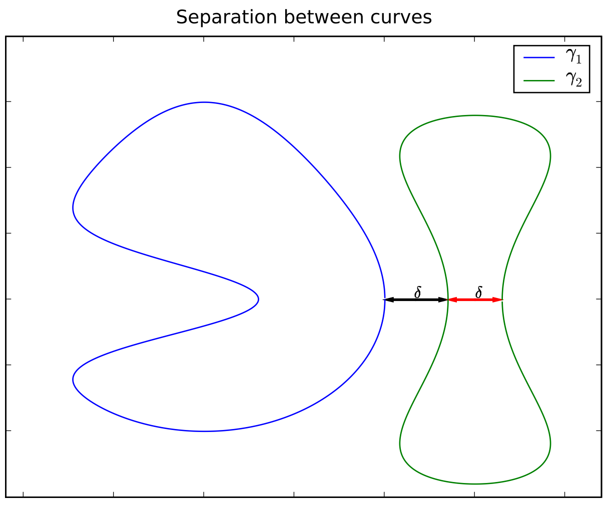

Figure 2: An illustration of Assumption 2. The black arrow illustrates the condition (1.2a), while the red arrow illustrates the condition (1.2b).

We also assume that different areas of the same curve are separated from each other:

$$\inf_{|t-t'|>\kappa_m{}^{-1}\pi/2} |\gamma_i(t) - \gamma_i(t')| \geq \delta \tag{1.2b}$$

(assuming the curve $\gamma_i(t)$ proceeds with unit speed).

These assumptions ensure that two distinct curves do not come too close together (1) and that separate regions of the same curve do not come arbitrarily close (2). This is illustrated in Figure 2.

2 The Reconstruction Algorithm

Before we begin, we require some notation.

Definition 2.1 *For a vector $\vec{v}$, let $\vec{v}^{\perp}$ denote the vector $\vec{v}$ rotated clockwise by an angle $\pi/2$.*

Definition 2.2 *Let $d(\vec{p},\vec{q})$ denote the usual Euclidean metric, $d(\vec{p},\vec{q}) = |\vec{p}-\vec{q}|$. Let $d_{\vec{m}}(\vec{p},\vec{q})$ denote the distance in the $\vec{m}$ direction between $\vec{p}$ and $\vec{q}$, i.e. $d_{\vec{m}}(\vec{p},\vec{q}) = |(\vec{p}-\vec{q}) \cdot \vec{m}|$.*

Definition 2.3 *For a point $\vec{p}$ and a curve $\gamma_i(t)$, we say that $\vec{p} \in \gamma_i(t)$ if $\exists t$ such that $\gamma_i(t) = \vec{p}$.*

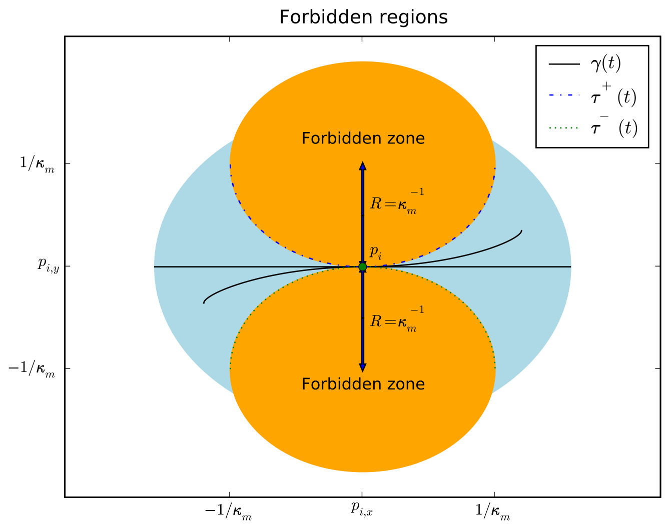

Figure 3: The forbidden zones, as described in Lemma 2.5. The orange (darker region) is the forbidden zone, and the blue (lighter region) is the set of points a distance $\pi{\kappa_m}^{-1}/2$ away from p_i.

2.1 The Forbidden Zone

Before explaining the algorithm which constructs the polygonalization of a figure (the set of curves $\{\gamma_i(t)\}_{0\ldots M-1}$) from discrete data $\{\vec{p}_i, \vec{m}_i\}_{i=0\ldots N-1}$, we prove a basic lemma which forms the foundation of our method. We assume for the remainder of this section that the figure satisfies Assumption 1.

Definition 2.4 *For a point $\vec{p}_i$, we refer to the set $\cup_{\pm} B_{{\kappa_m}^{-1}}(\vec{p}_i \pm \vec{m}_i^{\perp}{\kappa_m}^{-1})$ as its* forbidden zone*, illustrated in Fig. 3. Here, $B_r(\vec{p})$ is the usual ball of radius r about $\vec{p}$.*

Lemma 2.5 *For every $i \neq j$, if $\vec{p}_j$ is in the forbidden zone of $\vec{p}_i$, then $(\vec{p}_i, \vec{p}_j)$ is not an edge in Γ assuming that the sample spacing is less than ${\kappa_m}^{-1}\pi/2$.*

Proof. Suppose for simplicity that $\vec{p}_i = (0,0)$ and $\vec{m}_i = (1,0)$. Now, consider a line $\tau(t)$ of maximal curvature. The curve of maximal curvature, with $\tau_y'(t) > 0$ and proceeding at speed ${\kappa_m}^{-1}$ is $\tau^+(t) = ({\kappa_m}^{-1}\sin(t), {\kappa_m}^{-1}(1-\cos(t)))$, while the curve with $\tau_y'(t) < 0$ is $\tau^-(t) = ({\kappa_m}^{-1}\sin(t), {\kappa_m}^{-1}(\cos(t)-1))$.

By assumption 1, the curve $\gamma(t)$ containing $\vec{p}_i$ must lie between these curves (the near boundaries of the forbidden zone in Fig 2.5). Thus, it is confined to the blue (lighter) region while its arc length is less than ${\kappa_m}^{-1}\pi/2$. If $\vec{p}_j$ is in the forbidden zone and $\gamma(t)$ connects $\vec{p}_i$ to $\vec{p}_j$, then it must do so after travelling a distance greater than ${\kappa_m}^{-1}\pi/2$. $\square$

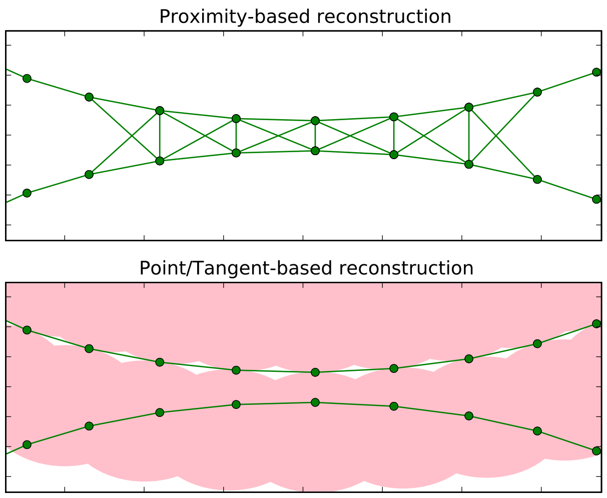
Proximity-based reconstruction
Point/Tangent-based reconstruction

Figure 4: A naive proximity-based reconstruction algorithm (shown), or even a β-crust type algorithm, will introduce edges between different curves. Knowledge of the forbidden zone allows us to remove such edges.

In short, the extra information provided by the tangents allows us to exclude edges from the polygonalization if they point too far away from the tangent, resulting in higher fidelity (c.f. Fig. 4).

Definition 2.6 *For a point $\vec{p}$, we define the* allowed zone *or* allowed region $A_\epsilon(\vec{p})$ *by*

$$A_\epsilon(\vec{p}) = B_\epsilon(\vec{p}_i) \setminus \left[\cup_{\pm} B_{{\kappa_m}^{-1}}(\vec{p}_i \pm \vec{m}_i^{\perp} {\kappa_m}^{-1})\right] \tag{2.1}$$

That is, $A_\epsilon(\vec{p})$ is the ball of radius ϵ about p excluding the forbidden zone.

Clearly, any edge in the polygonalization starting at $\vec{p}$, with length shorter than ϵ, must connect to another point $\vec{q} \in A_\epsilon(\vec{p})$. We are now ready to describe the polygonalization algorithm.

Algorithm 1 (Noise-Free Polygonalization)

Input: *[We assume we are given the dataset $\{\vec{p}_i, \vec{m}_i\}_{i=0\ldots N-1}$, the maximal curvature κ_m, and a parameter ϵ satisfying both $\epsilon\kappa_m < 1/\sqrt{2}$ and $2\kappa_m\epsilon^2 < \delta$. We assume that adjacent points on a given curve are less than a distance ϵ apart, i.e. the curve is ϵ-sampled.]*

1. *Compute the graph* $G = (\{\vec{p}_i\}_{i=0\ldots N-1}, E)$ *with edge set:*

$$E = \{(\vec{p}_i, \vec{p}_j) : \vec{p}_i \in A_\epsilon(\vec{p}_j) \text{ and } \vec{p}_j \in A_\epsilon(\vec{p}_i)\}$$

2. *For each vertex* $\vec{p}_i \in \{\vec{p}_i\}_{i=0\ldots N-1}$*:*

   a. *Compute the set of vertices*

$$R_i^\pm = \{\vec{p}_j : (\vec{p}_i, \vec{p}_j) \in E \text{ and } \pm(\vec{p}_j - \vec{p}_i) \cdot \vec{m}_i > 0\}$$

   b. *Find the nearest tangential neighbors, i.e.*

$$\vec{r}_i^\pm = argmin_{\vec{q} \in R_i^\pm} d_{\vec{m}_i}(\vec{q}, \vec{p}_i)$$

3. *Output the graph* $\Gamma = (\{\vec{p}_i\}_{i=0\ldots N-1}, E')$ *with*

$$E' = \{(\vec{p}_i, \vec{r}_i^+)\} \cup \{(\vec{p}_i, \vec{r}_i^-)\}$$

   This graph is the polygonalization of $\{\gamma_i(t)\}_{0\ldots M-1}$*.*

Remark 2.7 As presented, the complexity of Algorithm 4 is $O(N^2)$, due to both step 1 and step 2. (Step 2 can be slow if $O(N)$ points are within the allowed region of some particular point). The complexity can be reduced to $O(N \log N)$ using quadtrees if we assume a minimal sampling rate (see Appendix B).

The following theorem guarantees the correctness of Algorithm 4. Its proof is presented in the next section.

Theorem 2.8 *Suppose that:*

$$\delta > 2\kappa_m \epsilon^2 \tag{2.2a}$$

where δ *is as in Assumption 2 and also*

$$\epsilon < \frac{1}{\kappa_m \sqrt{2}} \tag{2.2b}$$

Suppose also that the distance between adjacent samples in the polygonalization is bounded by ϵ*, i.e. the curve is* ϵ*-sampled. Then graph* Γ *returned by Algorithm 4 is the polygonalization of* $\{\gamma_i(t)\}_{0\ldots M-1}$*.*

2.2 Proof of Theorem 2.8

Lemma 2.9 *Suppose* $i \neq j$ *and that Assumption 2 holds. Then for all* t, t'*, if* (2.2) *holds, then*

$$\gamma_j(t') \notin A_\epsilon(\gamma_i(t)). \tag{2.3}$$

Similarly, if $i = j$ *and* $|t - t'| \geq {\kappa_m}^{-1}\pi/2$*, then* (2.3) *holds.*

Proof. Fix t, and define $\vec{p} = \gamma_i(t)$ and $\vec{m} = \gamma_i'(t)/\left|\gamma_i'(t)\right|$. Define L to be the line segment $L = \{\vec{p} + \vec{m}{\kappa_m}^{-1}\sin(\theta) : \theta \in [-\arcsin(\epsilon\kappa_m), \arcsin(\epsilon\kappa_m)]\}$. The boundaries of $A_\epsilon(\vec{p})$ are given by

$$\vec{p} + \vec{m}{\kappa_m}^{-1}\sin(\theta) \pm \vec{m}^{\perp}{\kappa_m}^{-1}(1-\cos(\theta)).$$

Now, for any $\vec{q} \in \gamma_i$ and $\vec{q} \in A_\epsilon(\vec{p})$, the distance between $\vec{q}$ and L is the normal distance to L. This distance is bounded by:

$$\begin{aligned} d(\vec{q}, L) &\leq \sup_\theta {\kappa_m}^{-1}\left|(1-\cos(\theta))\right| \\ &\leq \sup_\theta {\kappa_m}^{-1} 2\sin^2(\theta/2) = 2{\kappa_m}^{-1}\sin^2(\arcsin(\epsilon\kappa_m)/2) \end{aligned} \tag{2.4}$$

The intermediate value theorem implies $\arcsin(x) \leq \arcsin'(\zeta)x = (1-\zeta^2)^{-1/2}x$ for some $\zeta \in [0, x]$; since $\epsilon\kappa_m < 2^{-1/2}$ (by (2.2b)), we find that:

$$\arcsin(\epsilon\kappa_m) \leq (1-(2^{-1/2})^2)^{-1/2}\kappa_m\epsilon = \sqrt{2}\kappa_m\epsilon$$

Substituting this into (2.4) yields:

$$d(\vec{q}, L) \leq 2{\kappa_m}^{-1}\sin^2(\sqrt{2}\kappa_m\epsilon/2) \leq \kappa_m\epsilon^2 \tag{2.5}$$

Thus, the *normal* distance between any point in $A_\epsilon(\vec{p})$ and L is $O(\kappa_m\epsilon^2)$.

If $\gamma_j(t') \notin L + \vec{m}^{\perp}\mathbb{R}$, then clearly $\gamma_j(t') \notin A_\epsilon(\gamma_i(t))$ so we assume $\gamma_j(t') \in L + \vec{m}^{\perp}\mathbb{R}$. In this case, $\gamma_j(t') = \vec{p} + \vec{m}{\kappa_m}^{-1}\sin(\theta_0) + \vec{m}^{\perp}z_j$ for some $\theta_0 \in [-\arcsin(\epsilon\kappa_m), \arcsin(\epsilon\kappa_m)]$ and $z_j \in \mathbb{R}$. Thus, $|z_j| = d_{\vec{m}^{\perp}}(\gamma_j(t'), L)$, the normal distance to L. By construction, there is a unique value t_i' such that $\gamma_i(t_i') = \vec{p} + \vec{m}{\kappa_m}^{-1}\sin(\theta_0) + \vec{m}^{\perp}z_i$. $|z_i|$ then equals $d_{\vec{m}^{\perp}}(\gamma_i(t), L)$. By the second triangle inequality,

$$d_{\vec{m}^{\perp}}(\gamma_j(t'), L) = |z_j| \geq \left||z_j - z_i| - |z_i|\right| \geq \delta - \kappa_m\epsilon^2 > \kappa_m\epsilon^2$$

But this implies that $d(\gamma_j(t'), L) \geq d_{\vec{m}^{\perp}}(\gamma_j(t'), L) \geq \kappa_m\epsilon^2$, and thus $\gamma_j(t') \notin A_\epsilon(\vec{p})$.

The proof when $i = j$ is identical. □

This result shows that the graph G, computed in Step 1 of Algorithm 4, separates different γ_i and γ_j from each other, as well as different parts of the same curve. Thus, after Step 1, we are left with a graph G having edges only between points $\vec{p}_i$ and $\vec{p}_j$ which are on the same curve γ_k, and which are separated along γ_k by an arc length no more than ${\kappa_m}^{-1}\pi/2$.

We now show that G is a superset of the polygonalization Γ.

Proposition 2.10 *Suppose the point data $\{\vec{p}_i\}_{i=0\ldots N-1}$ is ϵ-sampled, i.e. if two points $\vec{p}_i$ and $\vec{p}_j$ are adjacent on the curve γ_k, then the* arc length *between $\vec{p}_i$ and $\vec{p}_j$ is bounded by ϵ. Then G contains the polygonalization of $\{\gamma_i(t)\}_{0\ldots M-1}$.*

Proof. If the distance between adjacent points $\vec{p}_i$ and $\vec{p}_j$ is at most ϵ, then $\vec{p}_j \in B_\epsilon(\vec{p}_i)$. Since the segment of γ_k between $\vec{p}_i$ and $\vec{p}_j$ has arc length less than ϵ, $\vec{p}_j$ is not in the forbidden zone of $\vec{p}_i$ (by the same argument as in Lemma 2.5. Thus, $\vec{p}_j \in A_\epsilon(\vec{p}_i)$ (and vice versa), and $(\vec{p}_i, \vec{p}_j)$ is an edge in G. $\square$

We have now shown that G separates distinct curves, and that G contains the polygonalization Γ of $\{\gamma_i(t)\}_{0\ldots M-1}$. It remains to show that $G = \Gamma$.

Lemma 2.11 *A curve $\gamma_i(t)$ satisfying* (1.1) *admits the local parameterization*

$$\gamma_i(t) = \gamma(t_0) + (t - t_0)\gamma'(t_0) + w(t)\gamma'^{\perp}(t_0) \tag{2.6}$$

where $w'(t_0) = 0$. The parameterization is valid for $|t - t_0| < {\kappa_m}^{-1}$. In particular, $w(t) < f^{-1}(\kappa_m t)$ where $f(z) = z/\sqrt{1+z^2}$.

Proof. Taylor's theorem shows the parameterization to be valid on an arbitrarily small ball. All we need to do is show that this parameterization is valid on a region of size ${\kappa_m}^{-1}$.

The parameterization breaks down when $w'(t)$ blows up, so we need to show that this does not happen before $t = \epsilon$. Plugging this parameterization into the curvature bound (1.1) yields:

$$\frac{|w''(t)|}{(1 + w'(t)^2)^{3/2}} \leq \kappa_m$$

Assuming $w''(t)$ is positive, this is a first order nonlinear differential inequality for $w'(t)$. We can integrate both sides (using the hyperbolic trigonometric substitution $w(t) = \sinh(\theta)$ for the left side) to obtain:

$$\frac{w'(t)}{\sqrt{1 + w'(t)^2}} \leq \kappa_m t\,. \tag{2.7}$$

With $f(z)$ defined as in the statement, then $f^{-1}(z)$ is singular only at $z = \pm 1$, and is regular before that. Solving (2.7) for $w'(t)$ shows that:

$$w'(t) \leq f^{-1}(\kappa_m t)$$

implying that $w'(t)$ is finite for $\kappa_m t < 1$, or $t < {\kappa_m}^{-1}$. $\square$

Lemma 2.12 *Fix a point $\vec{p}_i = \vec{p} \in \{\vec{p}_i\}_{i=0\ldots N-1}$. Choose a tangent vector $\vec{m}_0$ and fix an orientation, say $+$. Consider the set of points $\vec{p}_j$ such that $(\vec{p}, \vec{p}_j)$ is an edge in G and $(\vec{p}_j - \vec{p}) \cdot \vec{m}_0 > 0$. Suppose also that ϵ satisfies* (2.2b)*. Then, the only edge in the polygonalization of γ is the edge for which $(\vec{p}_j - \vec{p}) \cdot \vec{m}_0$ is minimal.*

Proof. By Lemma 2.11, the curve $\gamma(t)$ can be locally parameterized as a graph near $\vec{p}$, i.e. (2.6). This is valid up to a distance ${\kappa_m}^{-1}$; by (2.2b), it is valid for all points in the graph G connected to $\vec{p}$.

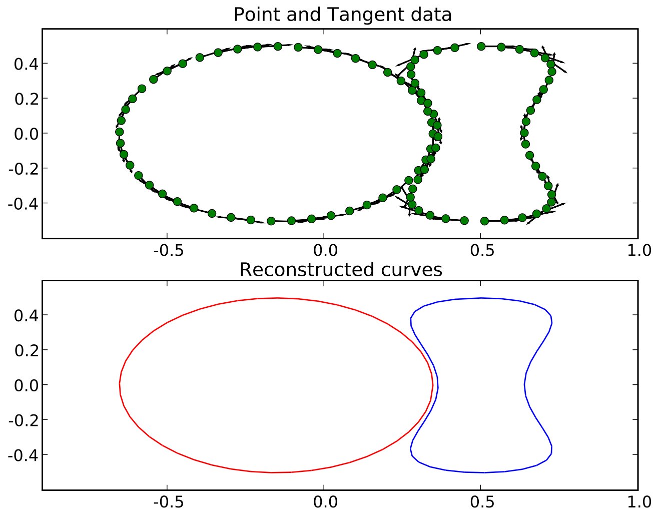

Figure 5: Some unordered points/tangents, and the curves reconstructed from them. In this case, $\epsilon = 0.065$, $\kappa_m = 3$ and $\delta = 0.015$.

The adjacent points on the graph are the ones for which $|t - t_0|$ is minimal. Note that $\vec{m}_0 \cdot (\vec{p}_j - \vec{p}) = t$ (simply plug in (2.6)); thus, minimizing $\vec{m}_0 \cdot (\vec{p}_j - \vec{p})$ selects the adjacent point on the graph. $\square$

The minimal edge is the edge $\vec{r}_0^{\,+}$ as computed in Step (2b) of Algorithm 4. Thus, we have shown that the computed graph G is the polygonalization Γ of $\{\gamma_i(t)\}_{0\ldots M-1}$.

3 Reconstruction in the Presence of Noise

In practice one rarely has perfect data, so it is important to understand the performance of the approach in the presence of errors. To that end, we consider the polygonalization problem, but with the point data perturbed by noise smaller than ζ and the tangent data perturbed by noise smaller than ξ. By this we mean the following; to each point $\vec{p}_i \in \{\vec{p}_i\}_{i=0\ldots N-1}$, there exists a point $\vec{p}_{i,*} = \gamma_{k_i}(t_i)$ such that $|\vec{p}_i - \vec{p}_{i,*}| \leq \zeta$. Similarly, the unit tangent vector $\vec{m}_i$ differs from the true tangent $\vec{m}_{i,*} = \gamma'_{k_i}(t_i)$ by an angle at most ξ. By a polygonalization of the noisy data, we mean that $(\vec{p}_i, \vec{p}_j)$ is an edge in the noisy polygonalization if $(\vec{p}_{i,*}, \vec{p}_{j,*})$ is an edge in the noise-free polygonalization. In what follows, $\vec{p}_j$ refers to a given (noisy) point, while $\vec{p}_{j,*}$ refers to the corresponding true point (and similarly for tangents).

Noise, of course, introduces a lower limit on the features we can resolve. At the very least, the curves must be separated by a distance greater than or equal to ζ, to prevent noise from actually moving a sample from one curve to another. In addition, noise in the tangent data introduces uncertainty which forces us to increase the sampling rate; in particular, we require $O(\epsilon\xi + \epsilon^2) < \delta$.

The main idea in extending Algorithm 4 to the noisy case is to expand the allowed regions to encompass all possible points and tangents. Of course, this imposes new constraints on the separation between curves.

We also require a *maximal* sampling rate in order to ensure that the order of points on the curve is not affected by noise. For work in the context of reconstruction using point samples only, see [7, 16].

Assumption 3 *We assume that adjacent points $\vec{p}_i$ and $\vec{p}_j$ on the curve $\gamma_k(t)$ are separated by a distance greater than $[(1 + 2^{3/2})(2\xi\epsilon + \zeta)]$.*

To compensate for noise, we expand the allowed region to account for uncertainty concerning the actual point locations.

Definition 3.1 *The* noisy allowed region $A_\epsilon^{\zeta,\xi}(\vec{p}_i)$ *is the union of the allowed regions of all points/tangents near $(\vec{p}_i, \vec{m}_i)$:*

$$A_\epsilon^{\zeta,\xi}(\vec{p}_i) = \bigcup_{\substack{|\vec{p}-\vec{p}_i|<\zeta \\ \arccos(\vec{m}_i\cdot\vec{m})<\xi}} \left(B_\epsilon(\vec{p}) \setminus \left[\cup_\pm B_{\kappa_m{}^{-1}}(\vec{p} \pm \vec{m}^\perp \kappa_m{}^{-1})\right]\right) \tag{3.1}$$

Algorithm 2 (Noisy Polygonalization)

Input: *[We assume we are given the dataset $\{\vec{p}_i, \vec{m}_i\}_{i=0\ldots N-1}$, the maximal curvature κ_m, the noise amplitudes ζ, ξ, and a parameter ϵ satisfying both $\epsilon\kappa_m < 1/\sqrt{2}$ and $4\zeta + 2\epsilon\xi + 2.1\kappa_m\epsilon^2 < \delta$. We assume that adjacent points on a given curve are less than a distance ϵ apart, i.e. the curve is ϵ-sampled.]*

1. *Compute the graph $G = (\{\vec{p}_i\}_{i=0\ldots N-1}, E)$ with edge set:*

$$E = \{(\vec{p}_i, \vec{p}_j) : B_\zeta(\vec{p}_i) \cap A_\epsilon^{\zeta,\xi}(\vec{p}_j) \neq \emptyset \text{ and } B_\zeta(\vec{p}_j) \cap A_\epsilon^{\zeta,\xi}(\vec{p}_i) \neq \emptyset\} \tag{3.2}$$

2. *For each vertex $\vec{p}_i \in \{\vec{p}_i\}_{i=0\ldots N-1}$:*

   a. *Compute the set of vertices*

$$R_i^\pm = \{\vec{p}_j : (\vec{p}_i, \vec{p}_j) \in E \text{ and } \pm(\vec{p}_j - \vec{p}_i)\cdot\vec{m}_i > 0\}$$

   b. *Find the nearest tangential neighbors, i.e.*

$$\vec{r}_i^\pm = argmin_{\vec{q}\in R_i^\pm} d_{\vec{m}_i}(\vec{q}, \vec{p}_i)$$

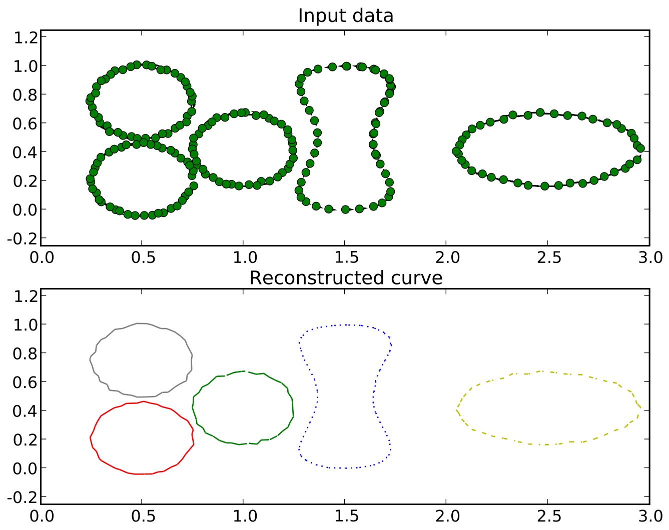

Figure 6: Noisy sampled points, and the reconstruction by Algorithm 2. This example takes $\kappa_m = 5$, $\epsilon = 0.15$, $\zeta = \xi = 0.01$.

3. *Output the graph* $\Gamma = (\{\vec{p}_i\}_{i=0\ldots N-1}, E')$ *with*

$$E' = \{(\vec{p}_i, \vec{r}_i^{+})\} \cup \{(\vec{p}_i, \vec{r}_i^{-})\}$$

This graph is the polygonalization of $\{\gamma_i(t)\}_{0\ldots M-1}$.

The following theorem guarantees that Algorithm 2 works. The proof follows that of Theorem 2.8 and is given in Appendix A. An application is shown in Fig. 6.

Theorem 3.2 *Suppose that Assumptions 1, 2 and 3 hold. Suppose also that*

$$\delta > 4\zeta + 4\epsilon\xi + 2.1\kappa_m\epsilon^2 \,, \tag{3.3a}$$

$$\epsilon < \frac{1}{\kappa_m\sqrt{2}} \,. \tag{3.3b}$$

Then, Algorithm 2 correctly reconstructs the figure.

Remark 3.3 Consider a point $\vec{p}$, which is a noisy sample from some curve in the figure. All we can say a priori is that $\vec{p}$ is close to the true sample $\vec{p}_*$, i.e. $\vec{p} \in B_\zeta(\vec{p}_*)$. However, given the knowledge that the polygonalization contains the edges $(\vec{q}, \vec{p})$ and $(\vec{p}, \vec{r})$, we can obtain further information on $\vec{p}_*$. Not only does $\vec{p}_*$ lie in $B_\zeta(\vec{p})$, but $\vec{p}_* \in A_\epsilon^{\zeta,\xi}(\vec{q})$ and $\vec{p}_* \in A_\epsilon^{\zeta,\xi}(\vec{r})$. In short,

$$\vec{p}_* \in B_\zeta(\vec{p}) \cap A_\epsilon^{\zeta,\xi}(\vec{q}) \cap A_\epsilon^{\zeta,\xi}(\vec{r}) \tag{3.4}$$

We can therefore improve our approximation to $\vec{p}_*$ by minimizing either the worst case error,

$$\vec{p}^{new} = \operatorname{argmin}_{\vec{p}} \sup_{\vec{x} \in A} |\vec{p} - \vec{x}| \, , \ A = B_\zeta(\vec{p}) \cap A_\epsilon^{\zeta,\xi}(\vec{q}) \cap A_\epsilon^{\zeta,\xi}(\vec{r}) \tag{3.5a}$$

or the mean error,

$$\vec{p}^{new} = \operatorname{argmin}_{\vec{p}} \int_A |\vec{p} - \vec{x}| \, d\vec{x} \tag{3.5b}$$

or some application-dependent functional. Noise in the tangential data can be similarly reduced. This is a postprocessing matter after polygonalization, and we will not expanded further on this idea in the present paper.

4 Examples

4.1 Extracting Topology from MRI images

In its simplest version, Magnetic Resonance Imaging (MRI) is used to obtain the two-dimensional Fourier transform of the proton density in a planar cross-section through the patient's body. That is, if $\rho(x)$ is is the proton density distribution in the plane P, then the MRI device is able to return the data $\hat{\rho}(k)$ at a selection of points k in the Fourier transform domain (k-space). The number of sample points available, however, is finite and covers only the low-frequency range in k-space well. Thus, it is desirable to be able to make use of the limited information in an optimal fashion. We are currently exploring methods for MRI based on exploiting the assumption that $\rho(x)$ is piecewise smooth (since different tissues have different densities, and the tissues boundaries tend to be sharp). Our goal is to carry out reconstruction in three steps. First, we find the tissue boundaries (the discontinu- ities). Second, we subtract the influence of the discontonuities from the measured k-space data and third, we reconstruct the remainder which is now smooth (or smoother). Standard filtered Discrete Fourier Transforms are easily able to reconstruct the remainder, so the basic problem is that of reconstructing the edges.

Using directional edge detectors on the k-space data, we can extract a set of point samples from the edges, together with non-oriented normal directions. By means of Algorithm 2, we can reconstruct the topology of the edge set and carry out the procedure sketched out above. The details of the algorithm are beyond the scope of this article, and will be reported at a later date, but Figure 7 illustrates the idea behind the method. Our work on curve reconstruction was, in fact, motivated by this application.

4.2 Figure detection

A natural problem in various computer vision applications is that of recognizing sampled objects that are partially obscured by a complex foreground. As a model of this problem, we constructed an (oval) figure, and obscured it by

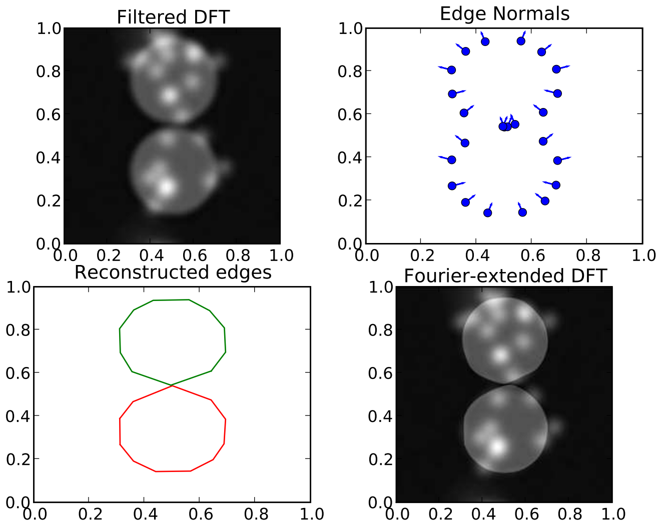

Figure 7: A simulated MRI image. The original image was two circles, together with some low frequency "texture". The noise level is 5%.

covering it with a sequence of curves. Algorithm 4 succesfully reconstructs the figure, as well as properly connecting points on the horizontally and vertically oriented covering curves. The result is shown in Figure 8. Note that the branches are not connected to the oval (or each other).

4.3 Filtering spurious points

The method provided here is relatively robust with regard to the addition of spurious random data points. This is because spurious data points are highly unlikely to be connected to any other points in the polygonalization graph. To see this, note first that for an incorrect data point to be connected to part of the polygonalization at all, it would need to be located in $A_\epsilon(\vec{p})$ for some $\vec{p}$. This is a region of length $O(\epsilon)$ and width $O(\epsilon^2)$. There are approximately $L = \sum_j \text{arclength}(\gamma_j)$ such points, for a total volume of $\epsilon^2 L$. Thus, the probability that a spurious point is in *some* allowed region is roughly $O(L\epsilon^2)$.

The second reason is that even if a spurious point is in some allowed region, it is unlikely to point in the correct direction. If an erroneous point $\vec{q}$ is inside $A_\epsilon(\vec{p})$, it is still not likely that $\vec{p} \in A_\epsilon(\vec{q})$, since the tangent at $\vec{q}$ must point in the direction of $\vec{p}$ (with error proportional to ϵ^2, the angular width of $A_\epsilon(\vec{q})$). Thus, the probability that the tangent at $\vec{q}$ points towards $\vec{p}$ is $O(\epsilon^2/2\pi)$. Combining these arguments, the probability that any *randomly chosen* spurious point $\vec{q}$ is connected to any other point in the polygonalization is $O(L\epsilon^4)$.

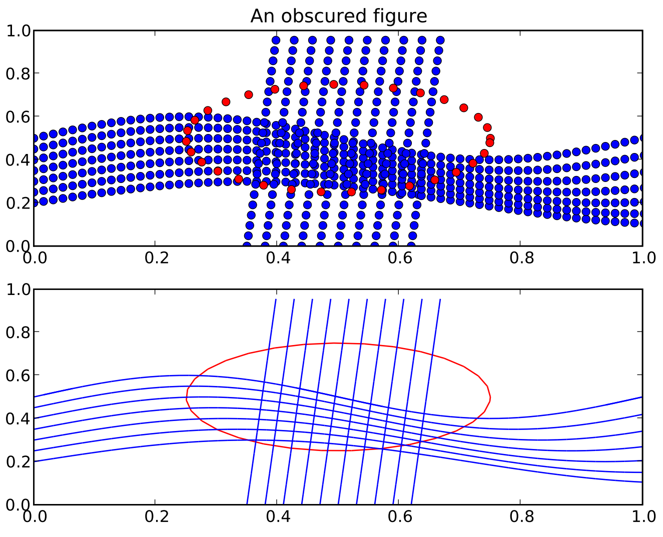

Figure 8: A figure which is partially obscured. Algorithm 4 correctly computes its polygonalization, and distinguishes it from the curves in front of it. To avoid visual clutter, the tangents are not displayed in this figure.

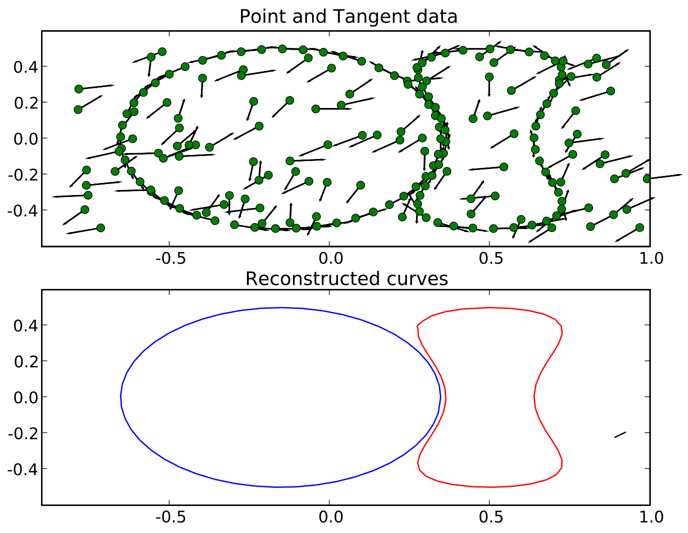

Figure 9: The same example as in Figure 5, but with 100 additional points (for a total of 196), placed randomly.

4.3.1 Filtering the data

The aforementioned criteria suggest that our reconstruction algorithm has excellent potential for noise removal. It suggests that if we remove points which do not have edges pointing towards other edges, then with high probability we are removing spurious edges.

This notion is well supported in practice. By running Algorithm 4 on a figure consisting of 96 true points, and 100 randomly placed incorrect points, a nearly correct polygonalization is calculated (Fig. 9). The original curve is reconstructed with an error at only one point (the top left corner of the right-hand curve).

Of course, if enough incorrect points are present, some points will eventually be connected by Algorithm 4. This can be seen in Figure 9: the line segment near $(0.9, -0.2)$ is an edge between two incorrect points. One hint that an edge is incorrect is that it points to a leaf. That is, consider a set of vertices $\vec{p}_0, \vec{p}_1, \ldots, \vec{p}_n$ as well as $\vec{q}$. Suppose, after approximately computing the polygonalization, one finds that the graph contains edges $e_0 = (\vec{p}_0, \vec{p}_1), e_1 = (\vec{p}_1, \vec{p}_2), \ldots, e_{n-1} = (\vec{p}_{n-1}, \vec{p}_n)$ and $e_n = (\vec{p}_{n/2}, \vec{q})$. The vertex $\vec{q}$ is a leaf, that is it is reachable by only one edge. A polygonalization of a set of closed curves should not have leaves, suggesting that the edge e_n is spurious. Thus filtering leaves is a very reasonable heuristic for noise filtering.

One final problem with noisy data worth mentioning is that sometimes, an incorrect point will be present that lies within the allowed region of a legitimate point, and closer to the legitimate point than the adjacent points along the curve. This will prevent the correct edge from being added. This can be remedied by adding not only $\vec{r}_i^{\pm}$ at Step 3 of the algorithm, but also points for which $d_{\vec{m}^\perp}(\vec{p}_i)$ whose distance to $\vec{p}_i$ is not much longer than the distance between $\vec{p}_i$ and $\vec{r}_i^{\pm}$. With some luck, this procedure combined with filtering out leaves will approximately reconstruct the correct figure.

Algorithm 3 (Polygonalization with Noise Removal)

Input:

[We assume we are given the dataset $\{\vec{p}_i, \vec{m}_i\}_{i=0\ldots N-1}$ (which includes spurious data), the maximal curvature κ_m, the noise amplitudes ζ, ξ, and a parameter ϵ satisfying both $\epsilon\kappa_m < 1/\sqrt{2}$ and $2\kappa_m\epsilon^2 < \delta$. We assume that adjacent points on a given curve are less than a distance ϵ apart, i.e. the curve is ϵ-sampled. We also assume we are given the number of leaf removal sweeps $l \in \mathbb{Z}^+$ and a threshold $\alpha \geq 1$.]

1. *Compute the graph $G = (\{\vec{p}_i\}_{i=0\ldots N-1}, E)$ with edge set:*

$$E = \{(\vec{p}_i, \vec{p}_j) : \vec{p}_i \in A_\epsilon(\vec{p}_j) \text{ and } \vec{p}_i \in A_\epsilon(\vec{p}_j)\}$$

2. *For each vertex $\vec{p}_i \in \{\vec{p}_i\}_{i=0\ldots N-1}$:*

a. *Compute the set of vertices*

$$R_i^{\pm} = \{\vec{p}_j : (\vec{p}_i, \vec{p}_j) \in E \text{ and } \pm(\vec{p}_j - \vec{p}_i) \cdot \vec{m}_i > 0\}$$

b. *Find the nearest tangential neighbors, i.e.*

$$\vec{r}_i^{\pm} = argmin_{\vec{q} \in R_i^{\pm}} \pm (\vec{p}_j - \vec{p}_i) \cdot \vec{m}_i$$

c. *Find the set of almost-nearest tangential neighbors:*

$$\mathbf{R}_i^{\pm} = \{\vec{r} \in R_i^{\pm} : d_{\vec{m}_i}(\vec{p}_i, \vec{r}) \leq \alpha \vec{r}_i^{\pm}\}$$

3. *Compute the graph* $\Gamma = (\{\vec{p}_i\}_{i=0\ldots N-1}, E')$ *with*

$$E' = \bigcup_i \{(\vec{p}_i, \vec{r}) : \vec{r} \in \mathbf{R}_i^{+}\} \cup \{(\vec{p}_i, \vec{r}) : \vec{r} \in \mathbf{R}_i^{-}\}$$

4. *Search through* Γ *for leaves, and remove edges pointing to the leaves. Repeat this* l *times.*

5. *Output* Γ.

In practice, we have found that $\alpha = 1.1$ and $l = 4$ work reasonably well. Figure 10 illustrates the result of Algorithm 3, both with and without filtering.

5 Conclusions

Standard methods for reconstructing a finite set of curves from sample data are quite general. By and large, they assume that only point samples are given. In some applications, however, additional information is available. In this paper, we have shown that if both sample location and tangent information are given, significant improvements can be made in accuracy. We were motivated by a problem in medical imaging, but believe that the methods developed here will be of use in a variety of other applications, including MR tractography and contour line reconstruction in topographic maps [13, 8].

A Proof of Theorem 3.2

The proof of Theorem 3.2 follows that of Theorem 2.8 closely, with minor modifications made to account for the noise. To begin, we need to show that the noisy allowed region is large enough to separate distinct curves. It is here that we use (3.3a).

Proposition A.1 *Suppose* $i \neq j$ *and assume that* (3.3) *holds. Then* $B_\zeta(\vec{p}_i) \cap A_\epsilon^{\zeta,\xi}(\vec{p}_j) = \emptyset$ *unless* $\vec{p}_i$ *and* $\vec{p}_j$ *are samples from the same curve, and are separated by an arc length no larger than* ${\kappa_m}^{-1}\pi/2$.

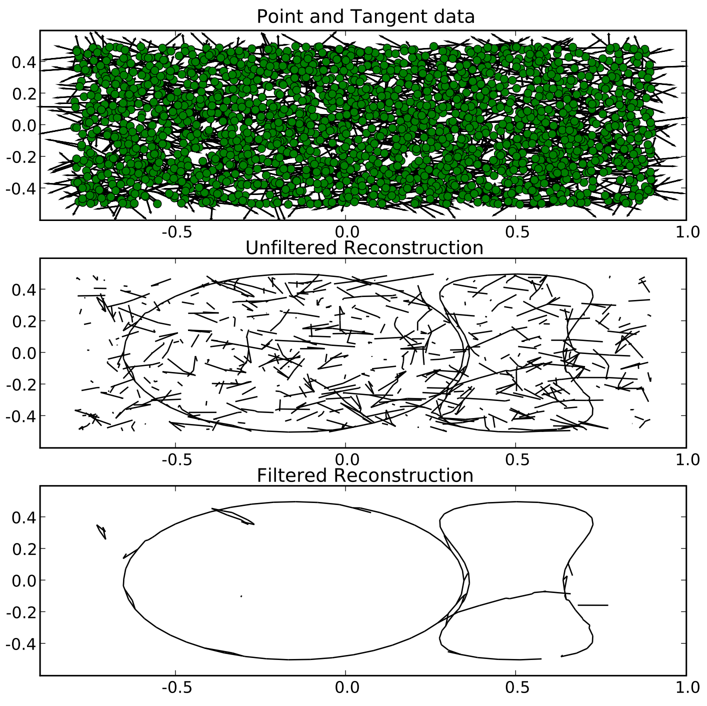

Figure 10: The same example as in Figure 5, but with 2000 additional random points added (for a total of 2096). The original curve is no longer completely reconstructed, but the general shape is still roughly visible, along with many more spurious points. The middle figure shows the reconstruction without Step 4 of Algorithm 3. Filtering leaves with $l = 4$ improves the situation considerably (bottom figure).

Proof. For simplicity, suppose that $\vec{p}_i \in B_\epsilon(\vec{p}_j)$ (since otherwise, $\vec{p}_i \notin A_\epsilon^{\zeta,\xi}(\vec{p}_j)$), but $\vec{p}_i$ is not sampled from the same curve as $\vec{p}_j$. Let $\gamma(t)$ denote the curve from which $\vec{p}_j$ is sampled. Let $\vec{p} = \vec{p}_j$ and $\vec{m} = \vec{m}_j$ to simplify notation.

Select points $\vec{p}'', \vec{m}''$ to minimize $d(\vec{p}_i, L'')$, where $L'' = \vec{p}'' + \vec{m}''$, with the constraint that $d(\vec{p}'', \vec{p}) < \zeta$ and the angle between $\vec{m}''$ and $\vec{m}$ is smaller than ξ. Let $\vec{a} \in L''$ be the point for which $d(\vec{a}, \vec{p}_i) = d(\vec{p}_i, L'')$.

It is shown in the proof of Lemma 2.9 that if $d(\vec{x}, L'') > \kappa_m \epsilon^2$, then $vx \notin A_\epsilon(\vec{p}'')$ (recall (2.4), (2.5)). Thus, if $d(\vec{p}_i, L'') > \kappa_m \epsilon^2 + \zeta$ for any $\vec{p}'', \vec{m}''$, then $\vec{x} \notin A_\epsilon(\vec{p}'')$ for each $\vec{x} \in B_\zeta(\vec{p}_i)$ and hence $\vec{p}_i \notin A_\epsilon^{\zeta,\xi}(\vec{p})$. We will show this to be the case.

By the second triangle inequality, we have the bound:

$$\begin{aligned} d(\vec{p}_i, L'') = d(\vec{p}_i, \vec{a}) &\geq d(\vec{p}_i', \vec{a}) - d(\vec{p}_i, \vec{p}_i') \\ &\geq d(\vec{p}_i', \vec{b}) - d(\vec{b}, \vec{a}) - d(\vec{p}_i, \vec{p}_i') \geq \delta - d(\vec{b}, \vec{a}) - \zeta \end{aligned} \tag{A.1}$$

where $\vec{b}$ is the point on $\gamma(t)$ closest to $\vec{p}_i'$. Once we show this is greater than $\kappa_m \epsilon^2$, the proof is complete.

Let $L' = \vec{p}' + \vec{m}'t$ (with $\vec{p}'$ and $\vec{m}'$ being true samples of $\gamma(t)$, approximated by $\vec{p}$ and vm). Then we have the bound:

$$d(\vec{b}, \vec{a}) \leq \sup_{\vec{g} \in L'} d(\vec{b}, \vec{g}) + d(\vec{g}, \vec{a}) \leq d(\vec{b}, L') + d(\vec{a}, L') \leq \kappa_m \epsilon^2 + d(\vec{a}, L') \tag{A.2}$$

The bound on $d(\vec{b}, \vec{g})$ follows since $\vec{b}$ is a sample from $\gamma(t)$ (recalling (2.4), (2.5)).

Since $\vec{a} = \vec{p}'' + \vec{m}''s$ (for some $s \in [-\epsilon, \epsilon]$), we can perform the bound:

$$\begin{aligned} d(\vec{a}, L') = \sup_{|t| \leq \epsilon} d(\vec{a}, \vec{p}' + \vec{m}'t) &\leq \sup_{|s| \leq \epsilon} \sup_{|t| \leq \epsilon} d(\vec{p}'' + \vec{m}''s, \vec{p}' + \vec{m}'t) \\ &\leq d(\vec{p}'' + m''\epsilon, \vec{p}' + \vec{m}'\epsilon) \leq 2\zeta + 4\xi\epsilon \end{aligned} \tag{A.3}$$

In (A.3), we assume m' and m'' are oriented the same way. It is easy enough to see that the sup is achieved at the endpoints; we then use the triangle inequality $d(\vec{p}'', vp') < d(\vec{p}'', \vec{p}) + d(\vec{p}, \vec{p}') \leq 2\zeta$, and similarly for the tangents. Thus, we find that:

$$\text{(A.2)} \leq \kappa_m \epsilon^2 + 2\zeta + 4\epsilon\xi \tag{A.4}$$

Plugging this into (A.1) shows that:

$$d(\vec{p}_i, L'') \geq \delta - (4\zeta + 4\epsilon\xi + \kappa_m \epsilon^2) \geq 1.1\kappa_m \epsilon^2 > \kappa_m \epsilon^2 + \zeta \tag{A.5}$$

where the last inequality follows from (3.3a). □

This shows that the graph G computed in Step 1 separates distinct curves.

The next result parallels Proposition 2.10, and shows that the noisy allowed region contains nearby points on the polygonalization.

Proposition A.2 *Suppose the figure is sampled at a rate satisfying* (2.2b). *Then G contains the polygonalization of the figure.*

Proof. The point $\vec{p}_i$ and tangent $\vec{m}_i$ are close to some point $\vec{p}'_i, \vec{m}'_i$ on the figure $\{\gamma_i(t)\}_{0\dots M-1}$; in particular, $|\vec{p}_i - \vec{p}'_i| \leq \zeta$ and $\arccos(\vec{m}_i \cdot \vec{m}'_i) < \xi$. Similarly, there is a point $\vec{p}'_j$ on the figure a distance no more than ζ away from $\vec{p}_j$. By Proposition 2.10, $\vec{p}'_j \in A_\epsilon(\vec{p}'_i)$. Since $\vec{p}'_j \in B_\zeta(\vec{p}_j)$ and $\vec{p}'_j \in A_\epsilon(\vec{p}'_i) \subset A^{\zeta,\xi}_\epsilon(\vec{p}_i)$, we find that $\vec{p}'_j \in B_\zeta(\vec{p}_j) \cap A^{\zeta,\xi}_\epsilon(\vec{p}_i) \neq \emptyset$. Repeating this argument with i and j interchanged shows that (3.2) holds, and $(\vec{p}_i, \vec{p}_j)$ is an edge of G. □

Proposition A.3 *Fix a point $\vec{p}_i = \vec{p} \in \{\vec{p}_i\}_{i=0\dots N-1}$, and suppose that Assumption 3 holds. Choose a tangent vector $\vec{m}_0$ and fix an orientation. Consider the set of points $\vec{p}_j$ such that $(\vec{p}, \vec{p}_j)$ is an edge in G (as per Step 1 of Algorithm 2) and $(\vec{p}_j - \vec{p}) \cdot \vec{m}_0 > 0$. Suppose also that ϵ satisfies* (3.3b).

Then the nearest tangential neighbor of $\vec{p}$ (i.e. the edge for which $(\vec{p}_j - \vec{p}) \cdot \vec{m}_0$ is minimal) is the edge in the polygonalization of γ.

Proof. The idea of the proof follows that of Lemma 2.12 closely, but we must adjust for our uncertainty as to the point and tangent.

The curve itself has the parameterization $\gamma(t) = \vec{p}' + \vec{m}'t + \vec{m}'^\perp w(t)$, by Lemma 2.11, and this is valid for $|t| < {\kappa_m}^{-1}$. However, we do not know $\vec{p}'$ and $\vec{m}'$, only $\vec{p}$ and $\vec{m}$. We wish to find the point $\vec{p}_j$ for which $\vec{m}' \cdot (\vec{p}'_j - \vec{p}')$ is minimal, and we approximate this by finding the point for which $\vec{m} \cdot (\vec{p}_j - \vec{p})$ is minimal.

Using the fact that $\vec{m} \cdot (\vec{p}_j - \vec{p}) - \vec{m} \cdot (\vec{p}_k - \vec{p}) = \vec{m} \cdot (\vec{p}_j - \vec{p}_k)$, we find that

$$\begin{aligned}\vec{m} \cdot (\vec{p}_j - \vec{p}) - \vec{m} \cdot (\vec{p}_k - \vec{p}) = \vec{m} \cdot (\vec{p}_j - \vec{p}_k) = \\ \vec{m} \cdot (\vec{p}'_j - \vec{p}'_k) + \vec{m} \cdot ([\vec{p}_j - \vec{p}'_j] - [\vec{p}_k - \vec{p}'_k]) \\ = \vec{m}' \cdot (\vec{p}'_j - \vec{p}'_k) + (\vec{m} - \vec{m}') \cdot (\vec{p}'_j - \vec{p}'_k) + \vec{m} \cdot ([\vec{p}_j - \vec{p}'_j] - [\vec{p}_k - \vec{p}'_k]) \end{aligned} \tag{A.6}$$

The second and third terms on the right side of (A.6) are the error terms. We have the bound:

$$\begin{aligned}&\left|(\vec{m} - \vec{m}') \cdot (\vec{p}'_j - \vec{p}'_k) + \vec{m} \cdot ([\vec{p}_j - \vec{p}'_j] - [\vec{p}_k - \vec{p}'_k])\right| \\ &\qquad \leq \sqrt{\sin^2(\xi) + (1 - \cos(\xi))^2}\left|\vec{p}'_j - \vec{p}'_k\right| + 2\zeta \leq 2(\xi\epsilon + \zeta)\end{aligned}$$

We wish to find the j for which (A.6) is negative for every k. If we can show that $\left|\vec{m}' \cdot (\vec{p}'_j - \vec{p}'_k)\right| > 2(\xi\epsilon + \zeta)$, we are done.

If we observe that $\vec{p}'_j = \vec{p}' + \vec{m}'t_j + w(t_j)\vec{m}'^\perp$ (using the notation of Lemma 2.11), and similarly $\vec{p}'_k = \vec{p}' + \vec{m}'t_k + w(t_k)\vec{m}'^\perp$, we find then that $\vec{m}' \cdot (\vec{p}'_j - \vec{p}'_k) = t_j - t_k$.

It is here we use the fact that $|\vec{p}_j - \vec{p}_k|^2 = (t_j - t_k)^2 + (w(t_j) - w(t_k))^2 \geq [(1 + 2^{3/2})(2\xi\epsilon + \zeta)]^2$. With $f(z)$ as in Lemma 2.11, we find that:

$$\begin{aligned}|(w(t_j) - w(t_k))| = |w'(y)|\,|(t_j - t_k)| &\leq \frac{1}{f'(f^{-1}(\kappa_m y))}\,|(t_j - t_k)| \\ &= (1 + (f^{-1}(\kappa_m y))^2)^{3/2}\,|(t_j - t_k)|\end{aligned}$$

for some $y \in [0, {\kappa_m}^{-1}]$. If $\kappa_m y < 1/\sqrt{2}$ (i.e. (2.2b) is satisfied), then $f^{-1}(\kappa_m y) < 1$ and $(1 + f^{-1}(\kappa_m y)^2) < 2$. Thus:

$$\begin{aligned} &\left|(1+2^{3/2})(t_j - t_k)\right|^2 \\ &\qquad \geq (t_j - t_k)^2 + [(1 + (f^{-1}(\kappa_m y))^2)^{3/2}(t_j - t_k)]^2 \\ &\qquad\qquad \geq |\vec{p}_j - \vec{p}_k|^2 \geq [(1+2^{3/2})(2\xi\epsilon + \zeta)]^2 \end{aligned} \tag{A.7}$$

(the last inequality follows from Assumption 3) implying that $|t_j - t_k| \geq 2(\xi\epsilon + \zeta)$. □

B Speeding it up: an $O(N \log N)$ algorithm

As remarked earlier, Algorithm 4 and 2 run in $O(N^2)$ time as written. The slow step is Step 1 which involves comparing every point/tangent pair to every other such pair. This scaling issue can be remedied by using a spatially adaptive data structure [12]

A caveat: there are two different ways of increasing N. The first (increasing outward) is by taking larger figures, with the sampling rate held fixed. The second (increasing inward) is by holding the figure size fixed, but increasing the sampling rate. We are interested primarily in the first case, and we will treat this case only. Therefore, we make the following additional assumption:

Assumption 4 *We assume that the density of points in the input data is bounded above, i.e.:*

$$\sup_B \frac{|\{\vec{p}_i\}_{i=0\ldots N-1} \cap B|}{|B|} \leq \rho_m \tag{B.1}$$

Note that this always holds in the case of noisy data. In this case, Assumption 3 combined with (3.3) implies that

$$\rho_m \leq \frac{1}{(1+2^{3/2})(2\xi+\zeta)\delta} \leq \frac{1}{(1+2^{3/2})(2\xi+\zeta)(4\zeta + 4\epsilon\xi + 2.1\kappa_m\epsilon^2)}.$$

In computing Step 1 of Algorithm 4 or 2, we must determine whether two points are in each other's allowed region (or a ball of radius ζ about the noisy allowed region). Note that $A_\epsilon(\vec{p}_i) \subset B_\epsilon(\vec{p}_i)$, so if $|\vec{p}_i - \vec{p}_j| \geq \epsilon$, then clearly the edge $(\vec{p}_i, \vec{p}_j) \notin G$. Similarly, for the noisy case, if $|\vec{p}_i - \vec{p}_j| \geq \epsilon + 2\zeta$, then $(\vec{p}_i, \vec{p}_j) \notin G$. We exploit this fact by using quadtrees, which allow us to avoid comparing points more than a distance ϵ apart.

Algorithm 4 (Fast Computation of the Graph G)

Input: *[We assume we are given the dataset* $\{\vec{p}_i, \vec{m}_i\}_{i=0...N-1}$*, the maximal point density* ρ_m *and the sampling* ϵ*. We also take the parameter* $\lambda = \epsilon$ *(noise free case) or* $\lambda = \epsilon + 2\zeta$ *(noisy case).]*

1. *Compute a quadtree* Q *storing* $(\vec{p}_i, \vec{m}_i)$ *pairs. The splitting criteria for a node is when the node contains more than* $\rho_m \lambda^2$ *points.*
2. *Initialize the graph* $G = (\{\vec{p}_i\}_{i=0...N-1}, E)$ *with empty edge set.*
3. *For each point* $\vec{p}_i$*, iterate over the points* $\vec{p}_j$ *contained in the node containing* $\vec{p}_i$ *and all of its nearest neighbors. If*

   $$\vec{p}_i \in A_\epsilon(\vec{p}_j) \text{ and } \vec{p}_j \in A_\epsilon(\vec{p}_i),$$

   then add the edge $(\vec{p}_i, \vec{p}_j)$ *to the graph* G*.*
4. *Return* G*.*

Initializing the quadtree in step 1 is an $O(N \log N)$ operation. Assumption 4 implies that the width of a node will be no smaller than λ; thus, a node containing a point $\vec{p}_i$ together with it's nearest neighbors contains the allowed region. The comparison at step 3 involves at most $\rho_m \lambda^2$ points, regardless of N. Thus, the complexity of this algorithm is

$$O(N \log(N) + N \rho_m \lambda^2). \tag{B.2}$$

References

[1] Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. *ACM Trans. Graph.*, 22(3):651–656, 2003.

[2] Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. Point-based computer graphics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, page 7, New York, NY, USA, 2004. ACM.

[3] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing: GMIP*, 60(2):125, 1998.

[4] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Computer Graphics*, 32(Annual Conference Series):415–421, 1998.

[5] Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1-2):125–141, 2002.

[6] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *Int. J. Comput. Vision*, 49(2-3):175–214, 2002.

[7] Siu-Wing Chen, Stefan Funke, Mordecai Golin, Piyush Kumar, Sheung-Hung Poon, and Edgar Ramos. Curve reconstruction from noisy samples. *Comput. Geometry*, 31:63–100, 2005.

[8] Y. Chen, R. Wang, and J. Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048–1057, 2006.

[9] Tamal K. Dey, Kurt Mehlhorn, and Edgar A. Ramos. Curve reconstruction: Connecting dots with good reason. In *Symposium on Computational Geometry*, pages 197–206, 1999.

[10] Tamal K. Dey and Rephael Wenger. Reconstructing curves with sharp corners. *Computational Geometry*, 19(2–3):89–99, 2001.

[11] Herbert Edelsbrunner. Shape reconstruction with Delaunay complex. In *LATIN '98: Theoretical Informatics*, Lecture Notes in Computer Science, 1380, Springer-Verlag, 119–32, 1998.

[12] Raphael Finkel and J.L. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys In *Acta Informatica*, 4(1): 1–9, 1974.

[13] A. Mishra, Y. Lu, A. S. Choe, A. Aldroubi, J. C. Gore, A, W. Andersona, Z. Ding. An image-processing toolset for diffusion tensor tractography. *Magnetic Resonance Imaging*, 25: 365–376, 2007.

[14] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[15] Thouis R. Jones, Fredo Durand, and Matthias Zwicker. Normal improvement for point rendering. *IEEE Comput. Graph. Appl.*, 24(4):53–56, 2004.

[16] Asisn Mukhopadhyay and Augustus Das. Curve reconstruction in the presence of noise. In *CGIV 2007: 4th International Conference on Computer Graphics, Imaging and Visualization*, pp. 177–182, IEEE Compuer Society, 2007.

[17] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[18] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, New York, NY, USA, 2001. ACM.